US009247062B2

(12) **United States Patent**
Fahlgren et al.

(10) **Patent No.: US 9,247,062 B2**
(45) **Date of Patent: Jan. 26, 2016**

(54) **SYSTEM AND METHOD FOR QUEUING A COMMUNICATION SESSION**

(71) Applicant: **Twilio, Inc.**, San Francisco, CA (US)

(72) Inventors: **Christer Fahlgren**, San Francisco, CA (US); **John Wolthuis**, San Francisco, CA (US); **Peter Shafton**, San Francisco, CA (US); **Thomas Schiavone**, San Francisco, CA (US)

(73) Assignee: **Twilio, Inc.**, San Francisco, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **13/921,941**

(22) Filed: **Jun. 19, 2013**

(65) **Prior Publication Data**

US 2013/0336472 A1 Dec. 19, 2013

Related U.S. Application Data

(60) Provisional application No. 61/661,730, filed on Jun. 19, 2012.

(51) **Int. Cl.**
H04M 3/00 (2006.01)
H04M 3/523 (2006.01)
(Continued)

(52) **U.S. Cl.**
CPC ***H04M 3/523*** (2013.01); *H04M 3/5141* (2013.01); *H04M 2203/407* (2013.01)

(58) **Field of Classification Search**
CPC . H04M 3/523; H04M 3/5191; H04M 3/5233; H04M 3/5232; H04M 3/51; H04M 3/5183; H04M 3/5238; H04M 3/5231; H04M 7/006; H04M 3/5307; H04M 2201/50; H04M 2203/4536; H04M 3/5166; H04M 2203/2011; H04M 3/5235; H04M 2203/551; H04M 3/4285; H04M 3/42195; H04M 3/428; H04M 3/42221; H04M 2203/407; H04M 3/5141; H04M 3/42068; H04M 3/5158; H04M 1/2473; H04M 3/367; H04Q 2213/13152; H04Q 2213/13053; H04Q 2213/13377; H04Q 3/64
USPC 379/202.01, 203.01, 204.01, 205.01, 379/206.01, 221, 261, 266, 210.01, 265.11, 379/266.07, 265.09, 266.01; 370/260, 261, 370/262, 352, 353, 354, 356, 357; 719/328, 719/314, 315; 455/416
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,274,700 A 12/1993 Gechter et al.
5,526,416 A 6/1996 Dezonno et al.
(Continued)

FOREIGN PATENT DOCUMENTS

DE 1684587 A 3/1971
EP 0282126 A 9/1988
(Continued)

OTHER PUBLICATIONS

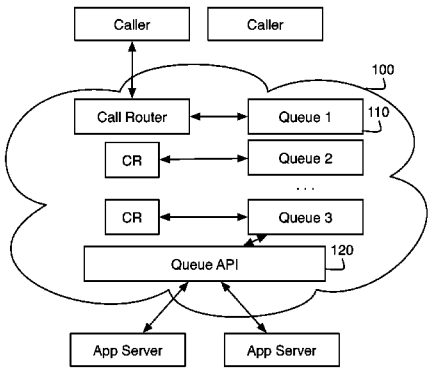
RFC 3986: Uniform Resource Identifier (URI): Generic Syntax; T. Berners-Lee, R. Fielding, L. Masinter; Jan. 2005; The Internet Society.*
(Continued)

Primary Examiner — Ahmad F. Matar
Assistant Examiner — Kharye Pope
(74) *Attorney, Agent, or Firm* — Jeffrey Schox

(57) **ABSTRACT**

A system and method including receiving a request to enqueue a communication session; adding the communication session to the queue of a plurality of communication sessions; upon adding the communication to the queue, transferring control logic to the configured wait-state application; receiving a dequeue request initiated by a second entity; in response to the dequeue request, managing the procession of communication sessions in the queue; and dequeuing a communication session from the queue.

31 Claims, 11 Drawing Sheets



(51) **Int. Cl.**

H04L 12/66	(2006.01)
H04M 3/51	(2006.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,581,608	A	12/1996	Jreij et al.	
5,598,457	A	1/1997	Foladare et al.	
5,934,181	A	8/1999	Adamczewski	
6,026,440	A	2/2000	Shrader et al.	
6,094,681	A	7/2000	Shaffer et al.	
6,138,143	A	10/2000	Gigliotti et al.	
6,185,565	B1	2/2001	Meubus et al.	
6,192,123	B1	2/2001	Grunsted et al.	
6,206,564	B1	3/2001	Adamczewski	
6,223,287	B1	4/2001	Douglas et al.	
6,269,336	B1	7/2001	Ladd et al.	
6,373,836	B1 *	4/2002	Deryugin et al.	370/352
6,425,012	B1	7/2002	Trovato et al.	
6,426,995	B1	7/2002	Kim et al.	
6,430,175	B1	8/2002	Echols et al.	
6,434,528	B1	8/2002	Sanders	
6,445,694	B1	9/2002	Swartz	
6,445,776	B1	9/2002	Shank et al.	
6,459,913	B2	10/2002	Cloutier	
6,493,558	B1	12/2002	Bernhart et al.	
6,496,500	B2	12/2002	Nance et al.	
6,501,832	B1	12/2002	Saylor et al.	
6,507,875	B1	1/2003	Mellen-Garnett et al.	
6,577,721	B1	6/2003	Vainio et al.	
6,600,736	B1	7/2003	Ball et al.	
6,606,596	B1	8/2003	Zirngibl et al.	
6,614,783	B1 *	9/2003	Sonesh et al.	370/352
6,625,258	B1	9/2003	Ram et al.	
6,625,576	B2	9/2003	Kochanski et al.	
6,636,504	B1	10/2003	Albers et al.	
6,662,231	B1	12/2003	Drosset et al.	
6,704,785	B1	3/2004	Koo et al.	
6,707,889	B1	3/2004	Saylor et al.	
6,711,249	B2	3/2004	Weissman et al.	
6,738,738	B2	5/2004	Henton	
6,757,365	B1	6/2004	Bogard	
6,765,997	B1	7/2004	Zirngibl et al.	
6,768,788	B1	7/2004	Langseth et al.	
6,778,653	B1	8/2004	Kallas et al.	
6,785,266	B2	8/2004	Swartz	
6,788,768	B1	9/2004	Saylor et al.	
6,792,086	B1	9/2004	Saylor et al.	
6,792,093	B2	9/2004	Barak et al.	
6,798,867	B1	9/2004	Zirngibl et al.	
6,807,529	B2	10/2004	Johnson et al.	
6,807,574	B1	10/2004	Partovi et al.	
6,819,667	B1	11/2004	Brusilovsky et al.	
6,820,260	B1 *	11/2004	Flockhart et al.	717/173
6,829,334	B1	12/2004	Zirngibl et al.	
6,834,265	B2	12/2004	Balasuriya	
6,836,537	B1	12/2004	Zirngibl et al.	
6,842,767	B1	1/2005	Partovi et al.	
6,850,603	B1	2/2005	Eberle et al.	
6,870,830	B1	3/2005	Schuster et al.	
6,873,952	B1	3/2005	Bailey et al.	
6,874,084	B1	3/2005	Dobner et al.	
6,885,737	B1	4/2005	Gao et al.	
6,888,929	B1	5/2005	Saylor et al.	
6,895,084	B1	5/2005	Saylor et al.	
6,898,567	B2	5/2005	Balasuriya	
6,912,581	B2	6/2005	Johnson et al.	
6,922,411	B1	7/2005	Taylor	
6,931,405	B2	8/2005	El-Shimi et al.	
6,937,699	B1	8/2005	Schuster et al.	
6,940,953	B1	9/2005	Eberle et al.	
6,941,268	B2	9/2005	Porter et al.	
6,947,417	B2	9/2005	Laursen et al.	
6,947,988	B1	9/2005	Saleh	
6,961,330	B1	11/2005	Cattan et al.	
6,964,012	B1	11/2005	Zirngibl et al.	
6,970,915	B1	11/2005	Partovi et al.	
6,977,992	B2	12/2005	Zirngibl et al.	
6,985,862	B2	1/2006	Stroem et al.	
6,999,576	B2	2/2006	Sacra	
7,003,464	B2	2/2006	Ferrans et al.	
7,006,606	B1	2/2006	Cohen et al.	
7,010,586	B1	3/2006	Allavarpu et al.	
7,020,685	B1	3/2006	Chen et al.	
7,039,165	B1	5/2006	Saylor et al.	
7,062,709	B2	6/2006	Cheung	
7,076,037	B1	7/2006	Gonen et al.	
7,076,428	B2	7/2006	Anastasakos et al.	
7,089,310	B1	8/2006	Ellerman et al.	
7,103,003	B2	9/2006	Brueckheimer et al.	
7,103,171	B1 *	9/2006	Annadata et al.	379/265.12
7,106,844	B1	9/2006	Holland	
7,111,163	B1	9/2006	Haney	
7,140,004	B1	11/2006	Kunins et al.	
7,143,039	B1	11/2006	Stifelman et al.	
7,197,331	B2	3/2007	Anastasakos et al.	
7,197,461	B1	3/2007	Eberle et al.	
7,197,462	B2	3/2007	Takagi et al.	
7,197,544	B2	3/2007	Wang et al.	
7,225,232	B2	5/2007	Elberse	
7,227,849	B1	6/2007	Raesaenen	
7,260,208	B2	8/2007	Cavalcanti	
7,266,181	B1	9/2007	Zirngibl et al.	
7,269,557	B1	9/2007	Bailey et al.	
7,272,212	B2	9/2007	Eberle et al.	
7,272,564	B2	9/2007	Phillips et al.	
7,277,851	B1	10/2007	Henton	
7,283,515	B2	10/2007	Fowler	
7,286,521	B1	10/2007	Jackson et al.	
7,287,248	B1	10/2007	Adeeb	
7,289,453	B2	10/2007	Riedel et al.	
7,296,739	B1	11/2007	Mo et al.	
7,298,732	B2	11/2007	Cho	
7,308,085	B2	12/2007	Weissman	
7,308,408	B1	12/2007	Stifelman et al.	
7,324,633	B2	1/2008	Gao et al.	
7,324,942	B1	1/2008	Mahowald et al.	
7,328,263	B1	2/2008	Sadjadi	
7,330,463	B1	2/2008	Bradd et al.	
7,330,890	B1	2/2008	Partovi et al.	
7,340,040	B1	3/2008	Saylor et al.	
7,349,714	B2	3/2008	Lee et al.	
7,369,865	B2	5/2008	Gabriel et al.	
7,373,660	B1	5/2008	Guichard et al.	
7,376,223	B2	5/2008	Taylor et al.	
7,376,586	B1	5/2008	Partovi et al.	
7,376,733	B2	5/2008	Connelly et al.	
7,376,740	B1	5/2008	Porter et al.	
7,412,525	B2	8/2008	Cafarella et al.	
7,428,302	B2	9/2008	Zirngibl et al.	
7,440,898	B1	10/2008	Eberle et al.	
7,447,299	B1	11/2008	Partovi et al.	
7,454,459	B1	11/2008	Kapoor et al.	
7,457,397	B1	11/2008	Saylor et al.	
7,473,872	B2	1/2009	Takimoto	
7,486,780	B2	2/2009	Zirngibl et al.	
7,496,054	B2	2/2009	Taylor	
7,500,249	B2	3/2009	Kampe et al.	
7,505,951	B2	3/2009	Thompson et al.	
7,519,359	B2	4/2009	Chiarulli et al.	
7,522,711	B1	4/2009	Stein et al.	
7,536,454	B2	5/2009	Balasuriya	
7,552,054	B1	6/2009	Stifelman et al.	
7,571,226	B1	8/2009	Partovi et al.	
7,613,287	B1	11/2009	Stifelman et al.	
7,623,648	B1	11/2009	Oppenheim et al.	
7,630,900	B1	12/2009	Strom	
7,631,310	B1	12/2009	Henzinger	
7,644,000	B1	1/2010	Strom	
7,657,433	B1	2/2010	Chang	
7,657,434	B2	2/2010	Thompson et al.	
7,668,157	B2	2/2010	Weintraub et al.	
7,672,295	B1	3/2010	Andhare et al.	
7,675,857	B1	3/2010	Chesson	
7,676,221	B2 *	3/2010	Roundtree et al.	455/414.1

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,715,547 B2 5/2010 Ibbotson et al.
7,779,065 B2 8/2010 Gupta et al.
7,875,836 B2 1/2011 Imura et al.
7,882,253 B2 2/2011 Pardo-Castellote et al.
7,920,866 B2 4/2011 Bosch et al.
7,926,099 B1 4/2011 Chakravarty et al.
7,936,867 B1 * 5/2011 Hill et al. 379/265.12
7,962,644 B1 6/2011 Ezerzer et al.
7,979,555 B2 7/2011 Rothstein et al.
8,023,425 B2 9/2011 Raleigh
8,069,096 B1 11/2011 Ballaro et al.
8,081,958 B2 12/2011 Soederstroem et al.
8,103,725 B2 1/2012 Gupta et al.
8,126,128 B1 2/2012 Hicks, III et al.
8,149,716 B2 4/2012 Ramanathan et al.
8,150,918 B1 4/2012 Edelman et al.
8,156,213 B1 4/2012 Deng et al.
8,196,133 B2 6/2012 Kakumani et al.
8,233,611 B1 * 7/2012 Zettner 379/266.07
8,243,889 B2 8/2012 Taylor et al.
8,266,327 B2 9/2012 Kumar et al.
8,295,272 B2 10/2012 Boni et al.
8,306,021 B2 11/2012 Lawson et al.
8,326,805 B1 12/2012 Arous et al.
8,346,630 B1 1/2013 McKeown
8,355,394 B2 1/2013 Taylor et al.
8,417,817 B1 4/2013 Jacobs
8,429,827 B1 4/2013 Wetzel
8,438,315 B1 5/2013 Tao et al.
8,462,670 B2 6/2013 Chien et al.
8,509,068 B2 8/2013 Begall et al.
8,532,686 B2 9/2013 Schmidt et al.
8,542,805 B2 9/2013 Agranovsky et al.
8,594,626 B1 11/2013 Woodson et al.
8,611,338 B2 12/2013 Lawson et al.
8,649,268 B2 2/2014 Lawson et al.
8,667,056 B1 3/2014 Proulx et al.
8,675,493 B2 3/2014 Buddhikot et al.
8,755,376 B2 6/2014 Lawson et al.
8,806,024 B1 8/2014 Francis et al.
8,837,465 B2 9/2014 Lawson et al.
8,838,707 B2 9/2014 Lawson et al.
9,014,664 B2 4/2015 Kim et al.
9,015,702 B2 4/2015 Bhat
2001/0038624 A1 11/2001 Greenberg et al.
2001/0043684 A1 11/2001 Guedalia et al.
2002/0006124 A1 1/2002 Jimenez et al.
2002/0006125 A1 1/2002 Josse et al.
2002/0006193 A1 * 1/2002 Rodenbusch et al. 379/266.01
2002/0067823 A1 * 6/2002 Walker et al. 379/266.01
2002/0077833 A1 6/2002 Arons et al.
2002/0126813 A1 9/2002 Partovi et al.
2002/0136391 A1 9/2002 Armstrong
2002/0165957 A1 11/2002 Devoe et al.
2002/0176378 A1 11/2002 Hamilton et al.
2002/0198941 A1 12/2002 Gavrilescu et al.
2003/0006137 A1 1/2003 Wei et al.
2003/0014665 A1 1/2003 Anderson et al.
2003/0018830 A1 * 1/2003 Chen et al. 709/328
2003/0026426 A1 2/2003 Wright et al.
2003/0046366 A1 3/2003 Pardikar et al.
2003/0051037 A1 3/2003 Sundaram et al.
2003/0058884 A1 3/2003 Kallner et al.
2003/0059020 A1 3/2003 Meyerson et al.
2003/0060188 A1 3/2003 Gidron et al.
2003/0061317 A1 3/2003 Brown et al.
2003/0061404 A1 3/2003 Atwal et al.
2003/0088421 A1 5/2003 Maes et al.
2003/0103620 A1 * 6/2003 Brown et al. 379/266.01
2003/0123640 A1 7/2003 Roelle et al.
2003/0195990 A1 10/2003 Greenblat
2003/0196076 A1 10/2003 Zabarski et al.
2003/0211842 A1 11/2003 Kempf et al.
2003/0231647 A1 * 12/2003 Petrovykh 370/429
2004/0011690 A1 1/2004 Marfino et al.
2004/0044953 A1 3/2004 Watkins et al.
2004/0052349 A1 3/2004 Creamer et al.
2004/0071275 A1 4/2004 Bowater et al.
2004/0101122 A1 5/2004 Da Palma et al.
2004/0102182 A1 5/2004 Reith et al.
2004/0165569 A1 8/2004 Sweatman et al.
2004/0172482 A1 9/2004 Weissman et al.
2004/0205689 A1 10/2004 Ellens et al.
2004/0213400 A1 10/2004 Golitsin et al.
2004/0218748 A1 11/2004 Fisher
2004/0228469 A1 11/2004 Andrews et al.
2004/0240649 A1 12/2004 Goel
2005/0005200 A1 1/2005 Matena et al.
2005/0010483 A1 1/2005 Ling
2005/0021626 A1 1/2005 Prajapat et al.
2005/0025303 A1 2/2005 Hostetler
2005/0038772 A1 2/2005 Colrain
2005/0043952 A1 2/2005 Sharma et al.
2005/0047579 A1 3/2005 Salame
2005/0091572 A1 4/2005 Gavrilescu et al.
2005/0125251 A1 6/2005 Berger et al.
2005/0128961 A1 * 6/2005 Miloslavsky et al. 370/270
2005/0135578 A1 6/2005 Ress et al.
2005/0141500 A1 6/2005 Bhandari et al.
2005/0147088 A1 * 7/2005 Bao et al. 370/352
2005/0177635 A1 8/2005 Schmidt et al.
2005/0181835 A1 8/2005 Lau et al.
2005/0228680 A1 10/2005 Malik
2005/0238153 A1 10/2005 Chevalier
2005/0240659 A1 10/2005 Taylor
2005/0243977 A1 11/2005 Creamer et al.
2005/0246176 A1 11/2005 Creamer et al.
2005/0289222 A1 12/2005 Sahim
2006/0008073 A1 * 1/2006 Yoshizawa et al. 379/266.07
2006/0015467 A1 1/2006 Morken et al.
2006/0047666 A1 3/2006 Bedi et al.
2006/0067506 A1 * 3/2006 Flockhart et al. 379/265.09
2006/0129638 A1 6/2006 Deakin
2006/0143007 A1 6/2006 Koh et al.
2006/0168334 A1 7/2006 Potti et al.
2006/0203979 A1 9/2006 Jennings
2006/0209695 A1 9/2006 Archer et al.
2006/0212865 A1 9/2006 Vincent et al.
2006/0215824 A1 9/2006 Mitby et al.
2006/0217823 A1 9/2006 Hussey
2006/0217978 A1 9/2006 Mitby et al.
2006/0222166 A1 10/2006 Ramakrishna et al.
2006/0256816 A1 11/2006 Yarlagadda et al.
2006/0262915 A1 11/2006 Marascio et al.
2006/0270386 A1 11/2006 Yu et al.
2006/0285489 A1 12/2006 Francisco et al.
2007/0002744 A1 1/2007 Mewhinney et al.
2007/0036143 A1 2/2007 Alt et al.
2007/0050306 A1 3/2007 Mcqueen
2007/0070906 A1 3/2007 Thakur
2007/0070980 A1 3/2007 Phelps et al.
2007/0071223 A1 * 3/2007 Lee et al. 379/265.02
2007/0074174 A1 3/2007 Thornton
2007/0121651 A1 5/2007 Casey et al.
2007/0127691 A1 * 6/2007 Lert, Jr. 379/265.05
2007/0127703 A1 6/2007 Siminoff
2007/0130260 A1 6/2007 Weintraub et al.
2007/0133771 A1 6/2007 Stifelman et al.
2007/0149166 A1 6/2007 Turcotte et al.
2007/0153711 A1 7/2007 Dykas et al.
2007/0167170 A1 7/2007 Fitchett et al.
2007/0192629 A1 8/2007 Saito
2007/0208862 A1 9/2007 Fox et al.
2007/0232284 A1 10/2007 Mason et al.
2007/0242626 A1 10/2007 Altberg et al.
2007/0265073 A1 11/2007 Novi et al.
2007/0286180 A1 * 12/2007 Marquette et al. 370/356
2007/0291905 A1 12/2007 Halliday et al.
2007/0293200 A1 * 12/2007 Roundtree et al. 455/414.1
2007/0295803 A1 12/2007 Levine et al.
2008/0005275 A1 1/2008 Overton et al.
2008/0025320 A1 1/2008 Bangalore et al.
2008/0037715 A1 2/2008 Prozeniuk et al.
2008/0037746 A1 2/2008 Dufrene et al.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2008/0040484 A1 2/2008 Yardley
2008/0052395 A1 2/2008 Wright et al.
2008/0091843 A1 4/2008 Kulkarni
2008/0101571 A1 5/2008 Harlow et al.
2008/0104348 A1 5/2008 Kabzinski et al.
2008/0134049 A1 6/2008 Gupta et al.
2008/0139166 A1 6/2008 Agarwal et al.
2008/0146268 A1 6/2008 Gandhi et al.
2008/0152101 A1 6/2008 Griggs
2008/0154601 A1 6/2008 Stifelman et al.
2008/0155029 A1 6/2008 Helbling et al.
2008/0162482 A1 7/2008 Ahern et al.
2008/0165708 A1 7/2008 Moore et al.
2008/0177883 A1 7/2008 Hanai et al.
2008/0209050 A1 8/2008 Li
2008/0222656 A1 9/2008 Lyman
2008/0232574 A1 9/2008 Baluja et al.
2008/0235230 A1 9/2008 Maes
2008/0256224 A1 10/2008 Kaji et al.
2008/0275741 A1 11/2008 Loeffen
2008/0310599 A1 12/2008 Purnadi et al.
2008/0313318 A1 12/2008 Vermeulen et al.
2008/0316931 A1 12/2008 Qiu et al.
2008/0317222 A1 12/2008 Griggs et al.
2008/0317232 A1 12/2008 Couse et al.
2008/0317233 A1 12/2008 Rey et al.
2009/0046838 A1* 2/2009 Andreasson 379/101.01
2009/0052437 A1 2/2009 Taylor et al.
2009/0052641 A1 2/2009 Taylor et al.
2009/0074159 A1 3/2009 Goldfarb et al.
2009/0075684 A1 3/2009 Cheng et al.
2009/0083155 A1 3/2009 Tudor et al.
2009/0089165 A1 4/2009 Sweeney
2009/0089699 A1 4/2009 Saha et al.
2009/0093250 A1 4/2009 Jackson et al.
2009/0125608 A1* 5/2009 Werth et al. 709/218
2009/0129573 A1 5/2009 Gavan et al.
2009/0136011 A1 5/2009 Goel
2009/0170496 A1 7/2009 Bourque
2009/0171659 A1 7/2009 Pearce et al.
2009/0171669 A1 7/2009 Engelsma et al.
2009/0171752 A1 7/2009 Galvin et al.
2009/0182896 A1 7/2009 Patterson et al.
2009/0217293 A1* 8/2009 Wolber et al. 719/313
2009/0220057 A1 9/2009 Waters
2009/0221310 A1 9/2009 Chen et al.
2009/0222341 A1 9/2009 Belwadi et al.
2009/0225748 A1 9/2009 Taylor
2009/0225763 A1 9/2009 Forsberg et al.
2009/0232289 A1 9/2009 Drucker et al.
2009/0235349 A1 9/2009 Lai et al.
2009/0252159 A1* 10/2009 Lawson et al. 370/352
2009/0276771 A1 11/2009 Nickolov et al.
2009/0288165 A1 11/2009 Qiu et al.
2009/0300194 A1 12/2009 Ogasawara
2009/0316687 A1* 12/2009 Kruppa 370/352
2009/0318112 A1 12/2009 Vasten
2010/0037204 A1 2/2010 Lin et al.
2010/0070424 A1 3/2010 Monk
2010/0082513 A1 4/2010 Liu
2010/0087215 A1 4/2010 Gu et al.
2010/0088187 A1 4/2010 Courtney et al.
2010/0088698 A1 4/2010 Krishnamurthy
2010/0115041 A1 5/2010 Hawkins et al.
2010/0142516 A1 6/2010 Lawson et al.
2010/0150139 A1 6/2010 Lawson et al.
2010/0167689 A1 7/2010 Sepehri-Nik et al.
2010/0188979 A1 7/2010 Thubert et al.
2010/0191915 A1 7/2010 Spencer
2010/0208881 A1 8/2010 Kawamura
2010/0217837 A1 8/2010 Ansari et al.
2010/0217982 A1 8/2010 Brown et al.
2010/0232594 A1* 9/2010 Lawson et al. 379/220.01
2010/0235539 A1 9/2010 Carter et al.
2010/0251329 A1 9/2010 Wei
2010/0251340 A1 9/2010 Martin et al.
2010/0281108 A1 11/2010 Cohen
2010/0291910 A1 11/2010 Sanding et al.
2011/0029882 A1 2/2011 Jaisinghani
2011/0053555 A1 3/2011 Cai et al.
2011/0078278 A1 3/2011 Cui et al.
2011/0081008 A1 4/2011 Lawson et al.
2011/0083179 A1 4/2011 Lawson et al.
2011/0093516 A1 4/2011 Geng et al.
2011/0096673 A1 4/2011 Stevenson et al.
2011/0110366 A1 5/2011 Moore et al.
2011/0131293 A1 6/2011 Mori
2011/0167172 A1 7/2011 Roach et al.
2011/0170505 A1 7/2011 Rajasekar et al.
2011/0176537 A1 7/2011 Lawson et al.
2011/0211679 A1 9/2011 Mezhibovsky et al.
2011/0251921 A1 10/2011 Kassaei et al.
2011/0253693 A1 10/2011 Lyons et al.
2011/0255675 A1 10/2011 Jasper et al.
2011/0265172 A1 10/2011 Sharma et al.
2011/0267985 A1 11/2011 Wilkinson et al.
2011/0274111 A1 11/2011 Narasappa et al.
2011/0276892 A1 11/2011 Jensen-Horne et al.
2011/0276951 A1 11/2011 Jain
2011/0280390 A1 11/2011 Lawson et al.
2011/0283259 A1 11/2011 Lawson et al.
2011/0289126 A1 11/2011 Aikas et al.
2011/0299672 A1 12/2011 Chiu et al.
2011/0310902 A1 12/2011 Xu
2011/0320449 A1 12/2011 Gudlavenkatasiva
2011/0320550 A1 12/2011 Lawson et al.
2012/0000903 A1 1/2012 Baarman et al.
2012/0011274 A1 1/2012 Moreman
2012/0017222 A1 1/2012 May
2012/0028602 A1 2/2012 Lisi et al.
2012/0036574 A1 2/2012 Heithcock et al.
2012/0039202 A1 2/2012 Song
2012/0079066 A1 3/2012 Li et al.
2012/0083266 A1 4/2012 VanSwol et al.
2012/0110564 A1 5/2012 Ran et al.
2012/0114112 A1* 5/2012 Rauschenberger et al. 379/265.02
2012/0149404 A1 6/2012 Beattie et al.
2012/0173610 A1 7/2012 Bleau et al.
2012/0174095 A1 7/2012 Natchadalingam et al.
2012/0198004 A1* 8/2012 Watte 709/206
2012/0201238 A1* 8/2012 Lawson et al. 370/352
2012/0208495 A1 8/2012 Lawson et al.
2012/0226579 A1 9/2012 Ha et al.
2012/0239757 A1 9/2012 Firstenberg et al.
2012/0254828 A1 10/2012 Aiylam et al.
2012/0281536 A1 11/2012 Gell et al.
2012/0288082 A1* 11/2012 Segall 379/266.07
2012/0290706 A1 11/2012 Lin et al.
2012/0304245 A1 11/2012 Lawson et al.
2012/0304275 A1 11/2012 Ji et al.
2012/0316809 A1 12/2012 Egolf et al.
2012/0321070 A1 12/2012 Smith et al.
2013/0029629 A1 1/2013 Lindholm et al.
2013/0031158 A1 1/2013 Salsburg
2013/0054684 A1 2/2013 Brazier et al.
2013/0058262 A1 3/2013 Parreira
2013/0067448 A1 3/2013 Sannidhanam et al.
2013/0156024 A1 6/2013 Burg
2013/0201909 A1 8/2013 Bosch et al.
2013/0204786 A1 8/2013 Mattes et al.
2013/0212603 A1 8/2013 Cooke et al.
2013/0244632 A1* 9/2013 Spence et al. 455/415
2014/0064467 A1 3/2014 Lawson et al.
2014/0105372 A1 4/2014 Nowack et al.
2014/0106704 A1 4/2014 Cooke et al.
2014/0123187 A1 5/2014 Reisman
2014/0129363 A1 5/2014 Lorah et al.
2014/0153565 A1 6/2014 Lawson et al.
2014/0185490 A1 7/2014 Holm et al.
2014/0254600 A1 9/2014 Shibata et al.
2014/0274086 A1 9/2014 Boerjesson et al.
2014/0282473 A1 9/2014 Saraf et al.
2014/0355600 A1 12/2014 Lawson et al.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2015/0004932	A1	1/2015	Kim et al.
2015/0004933	A1	1/2015	Kim et al.
2015/0023251	A1	1/2015	Giakoumelis et al.
2015/0181631	A1	6/2015	Lee et al.

FOREIGN PATENT DOCUMENTS

EP	1464418	A	10/2004
EP	1522922	A2	4/2005
EP	1770586	A1	4/2007
ES	2134107	A	9/1999
JP	10294788		4/1998
JP	2004166000	A	6/2004
JP	2004220118	A	8/2004
JP	2006319914	A	11/2006
WO	9732448	A	9/1997
WO	02087804		11/2002
WO	2006037492	A	4/2006
WO	2009018489	A	2/2009
WO	2009124223	A	10/2009
WO	2010037064	A	4/2010
WO	2010040010	A	4/2010
WO	2010101935	A	9/2010
WO	2011091085	A	7/2011

OTHER PUBLICATIONS

Complaint for Patent Infringement, Telinit Technologies, *LLC* v.*Twilio Inc.*, dated Oct. 12, 2012.

* cited by examiner

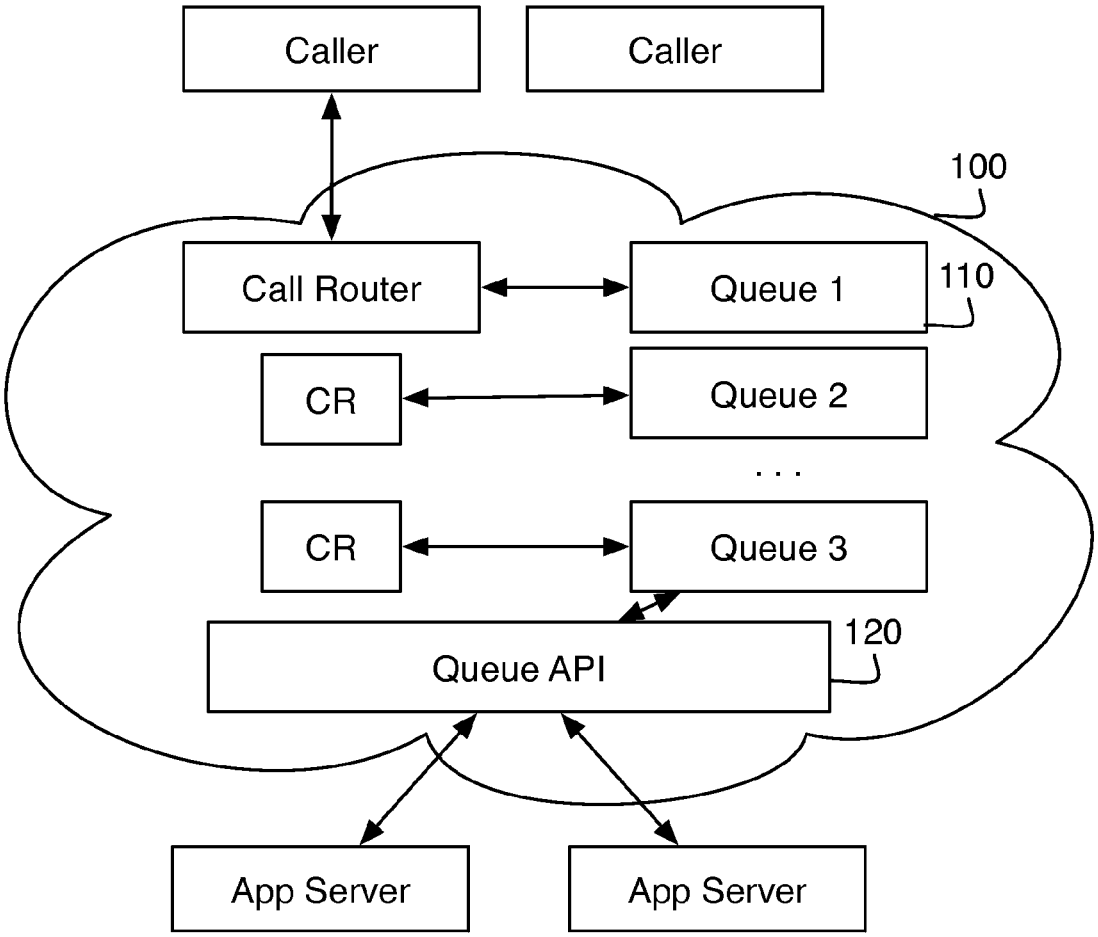

Caller
Caller
100
Call Router
Queue 1
110
CR
Queue 2
. . .
CR
Queue 3
120
Queue API
App Server
App Server

FIGURE 1

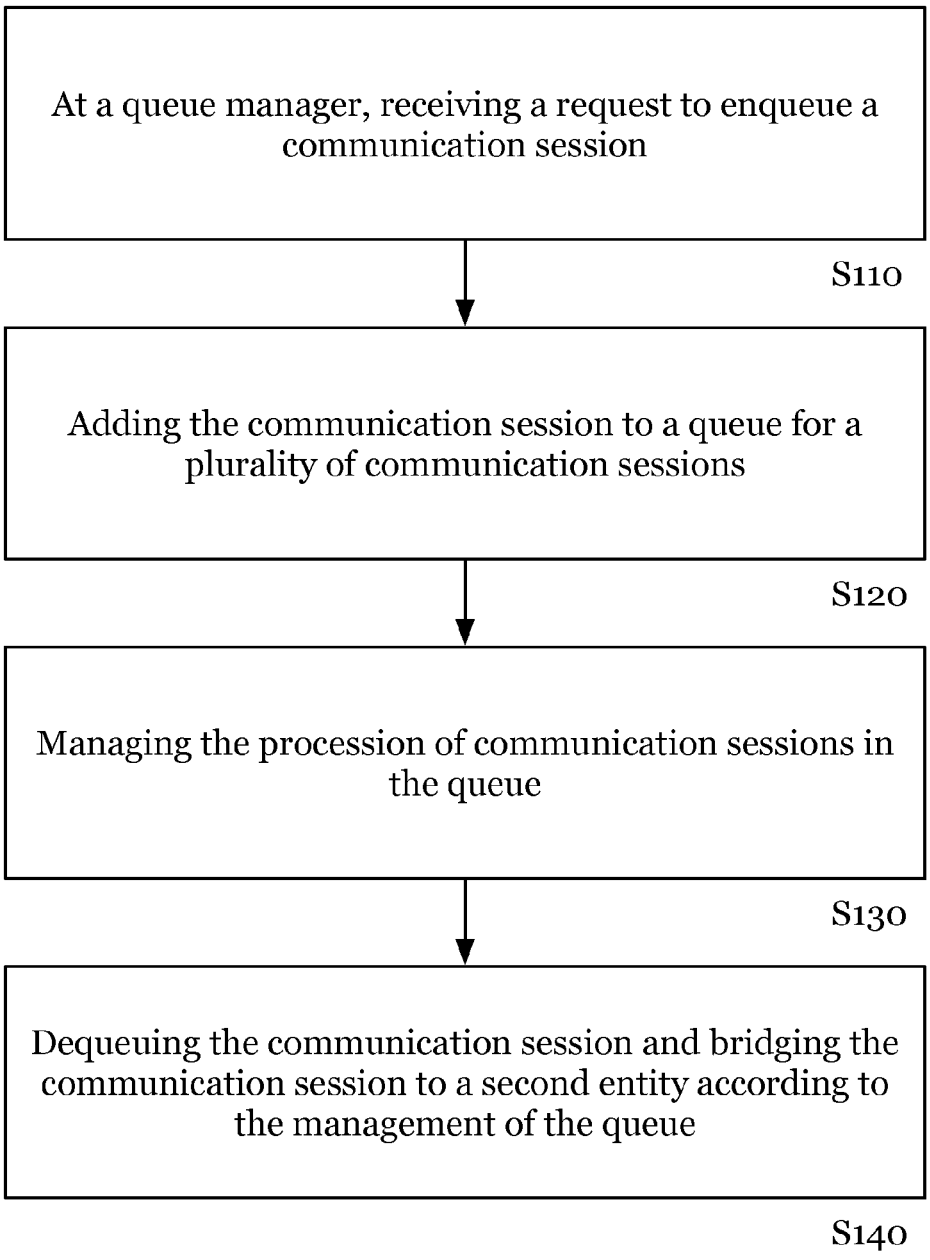

At a queue manager, receiving a request to enqueue a communication session
S110
Adding the communication session to a queue for a plurality of communication sessions
S120
Managing the procession of communication sessions in the queue
S130
Dequeuing the communication session and bridging the communication session to a second entity according to the management of the queue
S140

FIGURE 2

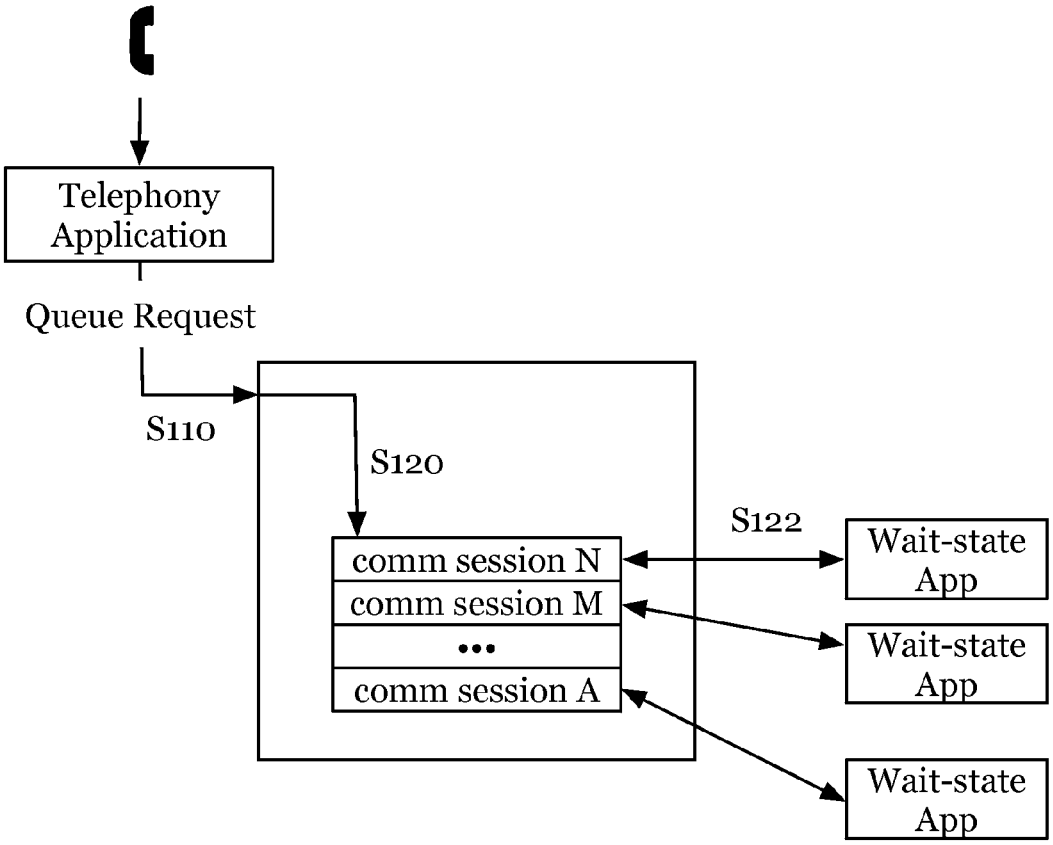

Telephony Application
Queue Request
S110
S120
S122
comm session N
comm session M
•••
comm session A
Wait-state App
Wait-state App
Wait-state App

FIGURE 3A

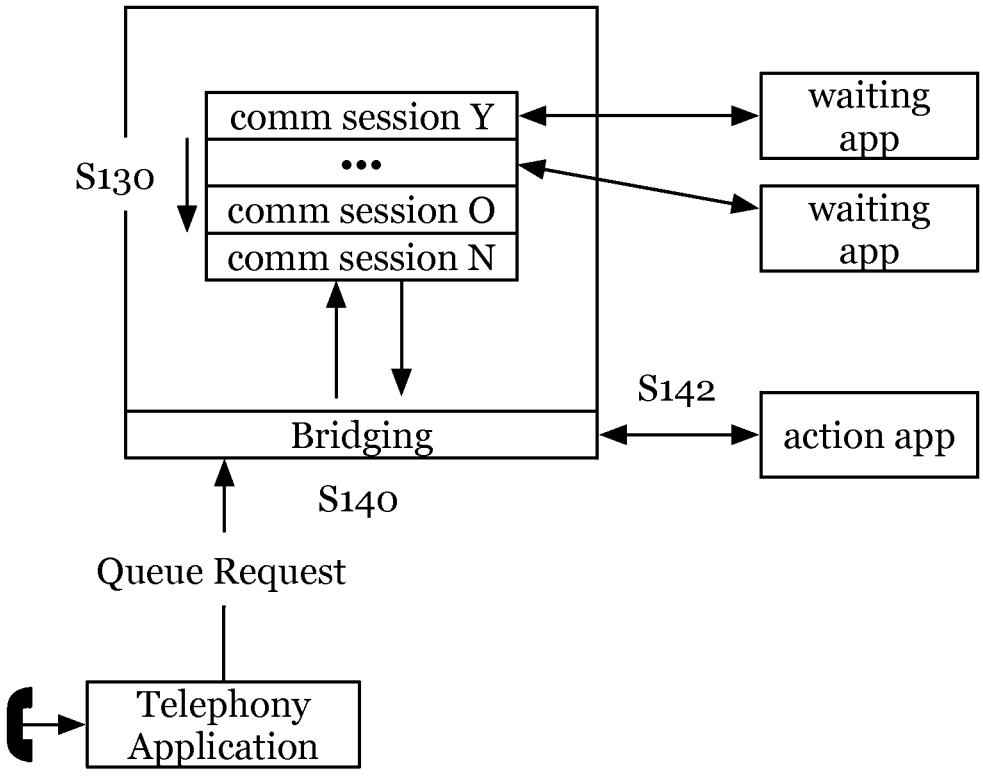

comm session Y
•••
comm session O
comm session N
S130
waiting app
waiting app
S142
Bridging
action app
S140
Queue Request
Telephony Application

FIGURE 3B

```
<Response>
  <Say>
    Please hold, a customer service representative will be with you shortly
  </Say>
  <Enqueue
      action="after_verb_exit.php"
      method="POST"
      waitUrl="hold_music.php"
      waitUrlMethod="GET">
      queue_name
  </Enqueue>
</Response>
```

FIGURE 4

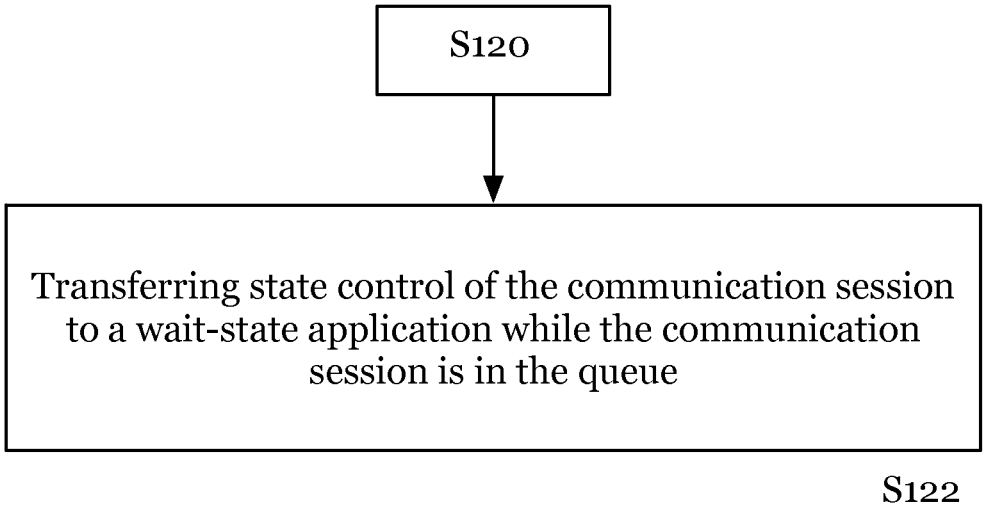

S120
Transferring state control of the communication session to a wait-state application while the communication session is in the queue
S122

FIGURE 5

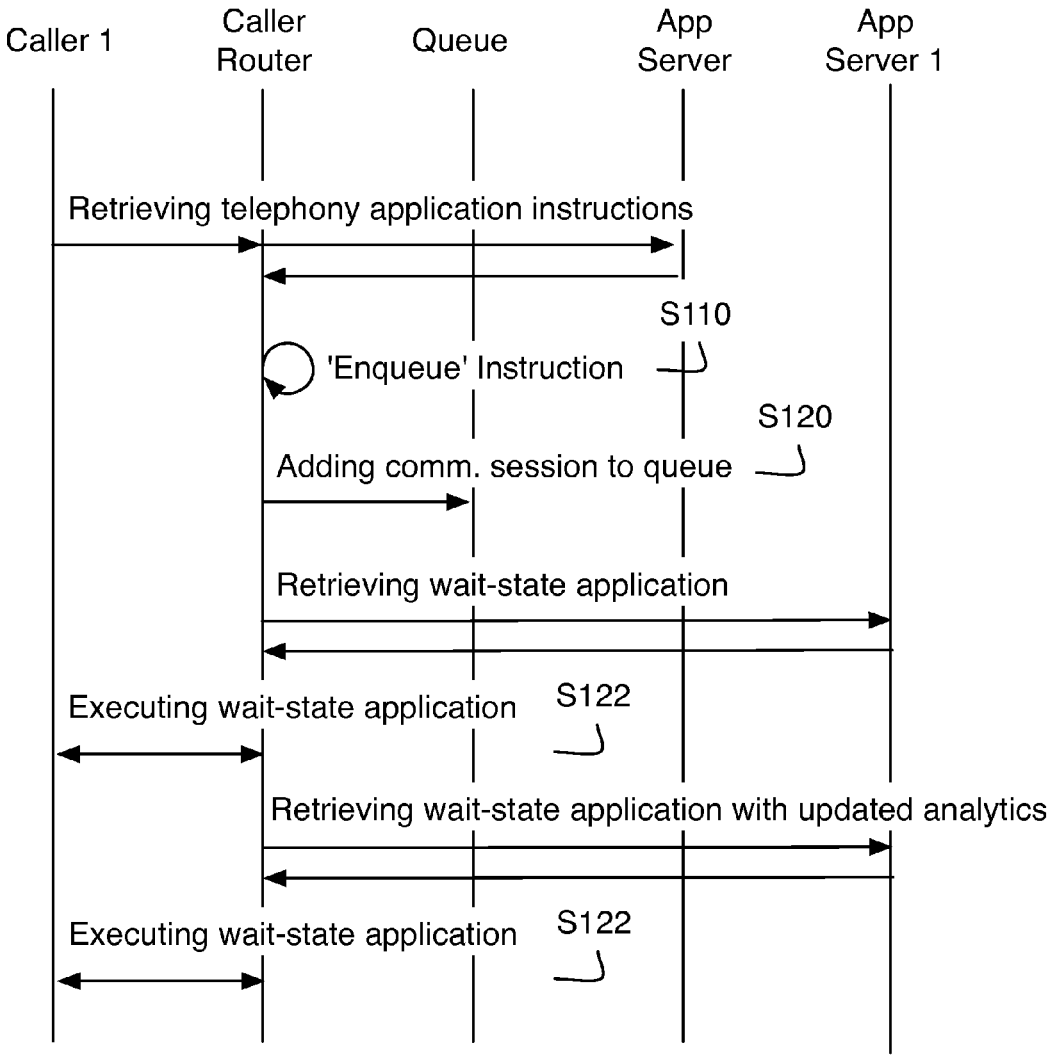

Caller 1
Caller Router
Queue
App Server
App Server 1
Retrieving telephony application instructions
S110
'Enqueue' Instruction
S120
Adding comm. session to queue
Retrieving wait-state application
Executing wait-state application
S122
Retrieving wait-state application with updated analytics
Executing wait-state application
S122

FIGURE 6

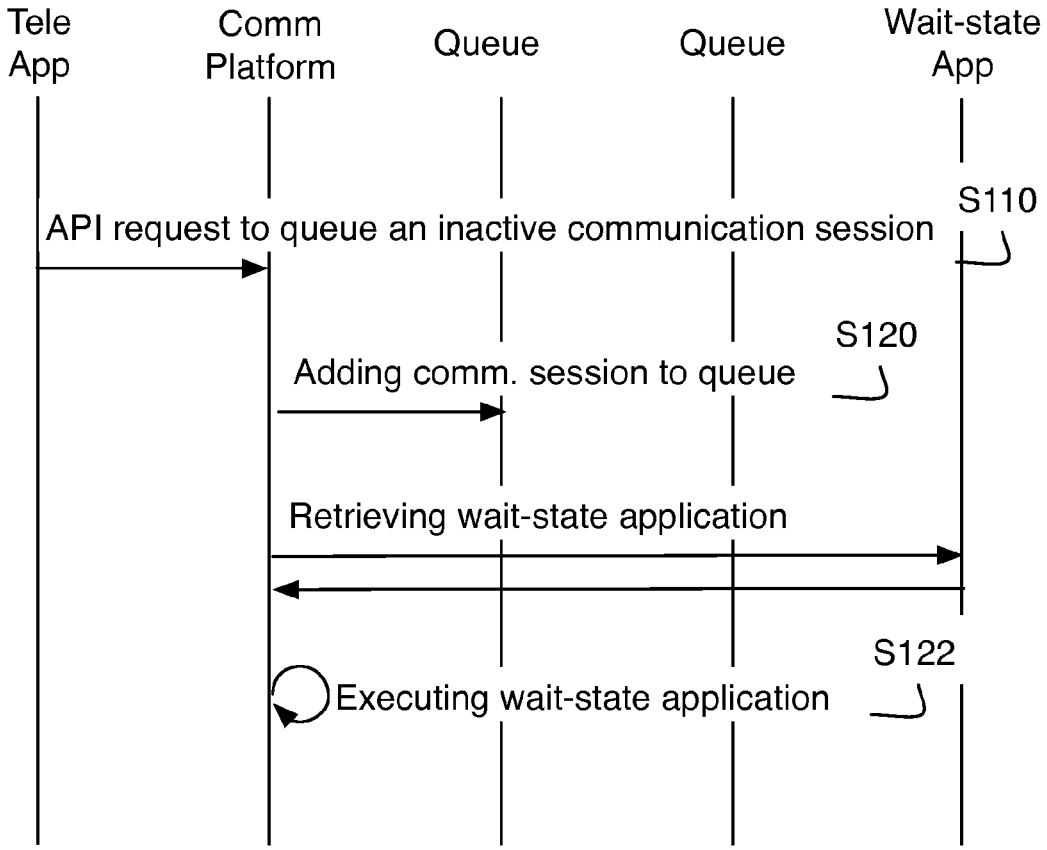

Tele App
Comm Platform
Queue
Queue
Wait-state App
API request to queue an inactive communication session
S110
Adding comm. session to queue
S120
Retrieving wait-state application
Executing wait-state application
S122

FIGURE 7

```
<Response>
  <Dial>
    <Queue>queue_name</Queue>
  </Dial>
</Response>
```

FIGURE 8

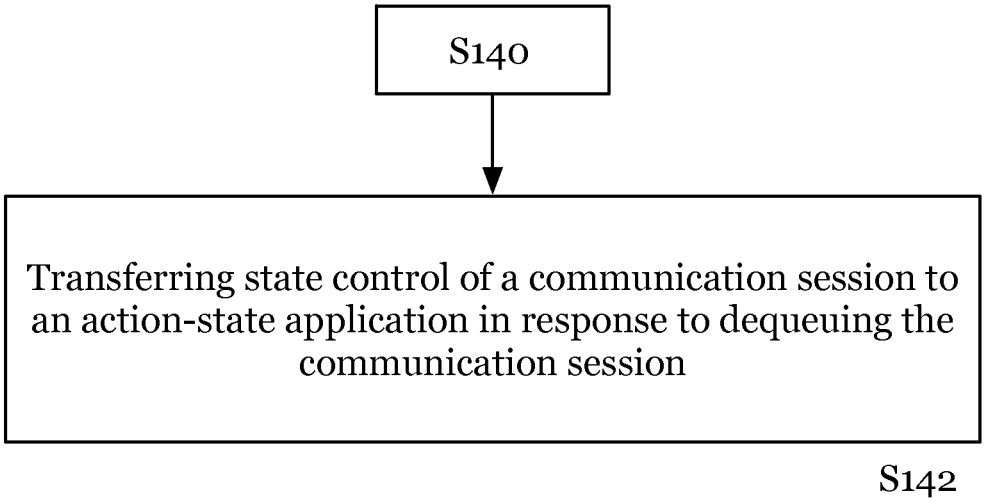

S140
Transferring state control of a communication session to an action-state application in response to dequeuing the communication session
S142

FIGURE 9

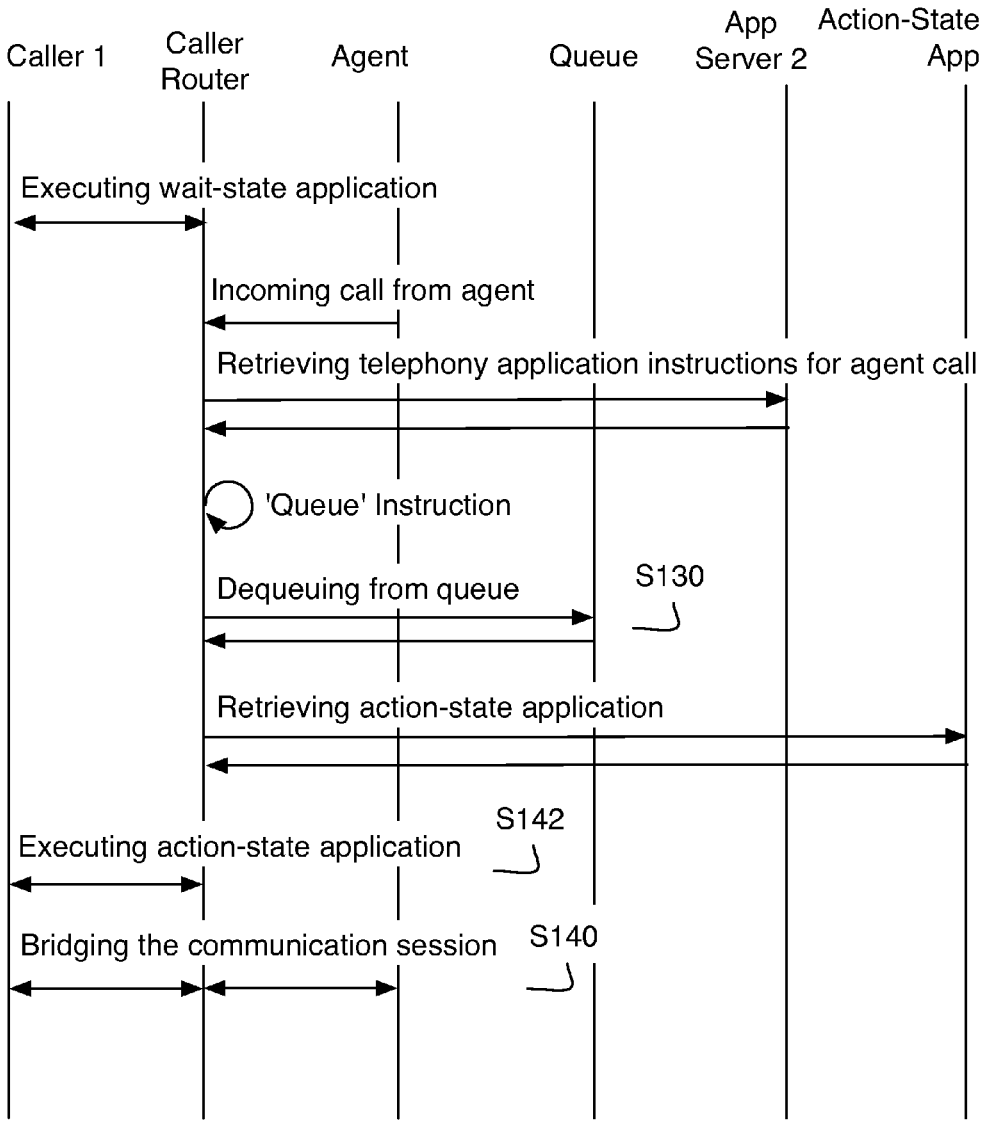

Caller 1
Caller Router
Agent
Queue
App Server 2
Action-State App
Executing wait-state application
Incoming call from agent
Retrieving telephony application instructions for agent call
'Queue' Instruction
Dequeuing from queue
S130
Retrieving action-state application
S142
Executing action-state application
Bridging the communication session
S140

FIGURE 10

SYSTEM AND METHOD FOR QUEUING A COMMUNICATION SESSION

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application Ser. No. 61/661,730, filed on 19 Jun. 2012, which is incorporated in its entirety by this reference.

TECHNICAL FIELD

This invention relates generally to the telephony field, and more specifically to a new and useful system and method for queuing a communication session in the telephony field.

BACKGROUND

In recent years, telephony applications have seen advancement due in part to internet based interfaces for telephony applications. The application of internet-based technologies in telephony communication has dramatically expanded the possibilities for telephony applications. Voice or telephony based calls are often limited by the fact that often a user is on one end of the call. In many cases, the resources to support these callers, such as customer service representatives, cannot be dynamically scaled to meet demand. As a result, telephony applications place callers in a holding pattern with the familiar looping music and canned messages. Waiting to leave the holding pattern is an annoyance to the caller, and the bad user experience reflects poorly upon the operators of the telephony application. Additionally, a generic waiting experience will not be suitable for all applications using a telephony platform that supports a wide variety of applications. Some PBX solutions provide basic static customization of music and announcements, but fail to provide flexibility use the queue. Thus, there is a need in the telephony field to create a new and useful system and method for queuing a communication session. This invention provides such a new and useful system and method.

BRIEF DESCRIPTION OF THE FIGURES

FIG. **1** is a schematic representation of a system of a preferred embodiment;

FIG. **2** is a flowchart representation of a first preferred embodiment of the invention;

FIGS. **3**A and **3**B are schematic representations of a preferred embodiment of the invention;

FIG. **4** is a exemplary request to enqueue a communication session;

FIG. **5** is a flowchart depicting a variation of the first preferred method for queuing a communication session;

FIG. **6** is a communication flowchart representation of a communication session being enqueued;

FIG. **7** is a communication flowchart representation of an inactive communication session being enqueued;

FIG. **8** is an exemplary request to dial a queue;

FIG. **9** is a flowchart depicting another variation of the first preferred method for queuing a communication session; and

FIG. **10** is a communication flowchart representation of dequeuing a communication session.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The following description of the preferred embodiments of the invention is not intended to limit the invention to these preferred embodiments, but rather to enable any person skilled in the art to make and use this invention.

As shown in FIG. **1**, a system for queuing a communication session of a preferred embodiment includes a multitenant communication platform **100**, a queue management resource no, and an API **120** with queue targeted interfaces. The system functions to enable programmatic and persistent queue resources. The system provides a platform tool wherein concurrent communication sessions can be simply managed while enabling advanced interaction capabilities. When implemented along with synchronous communication (e.g., telephone calls, video chats, screen-sharing, etc.), the system can be used as a tool for "call waiting". The system may additionally or alternatively be implemented for programmatic control of a queue for limited resources such as specialized processors, human facilitated processes, manufacturing tools, limited access resources, and/or any suitable resource limited component. The system preferably enables applications to simply direct inbound calls to a queue of their selection without implementing highly specialized queue application logic. The system is preferably provided to developers as a resource primitive that simplifies queue interactions and enables users of the platform (e.g., telephony application developers) to flexibly design customized and dynamic interactions. The system further functions to enable queues to be customized and interacted with dynamically and on the fly. Rich interactions and highly customized wait experiences can be generated by wait-state applications. When in the queue, the experience can be specifically tailored for the user through customized applications. Once an application is ready to connect a caller, a communication session can be easily popped off the queue and, using application instructions or an API, the communication session is directed to the appropriate place.

In an exemplary implementation, the system allows accounts or different applications to establish various call queues. The call queues can be for any suitable resource. In many cases, the call queue will be for a connection to an available agent (e.g., customer service representative). In other cases, the call queue could be to a limited resource such as a processing server that provides some unique service such as image processing. Communication sessions can preferably be enqueued during an active communication (e.g., an entity is concurrently participating in a live communication session) or in preparation for an eventually active communication session (e.g., a proxy for an upcoming call is enqueued and a live communication session is not active while waiting in the queue). A communication session may be enqueued in response to an API request and/or an instruction of an application. Any suitable number of queue resources can be generated for an account, and the platform can enable any suitable number of accounts to maintain queue resources. For example, an account on the platform may use the platform to provide a customer service phone system. There may be three call queues that are used for three different regions of customer support. Simultaneously, a second account may use a different queue setup to support a different application. The queues of the preferred embodiment are additionally configured for application execution during various states of waiting in the queue. Different applications, instructions, media resources or other resources can be set to be executed/played for a communication session for a particular state in the call queue. These various queue-state applications (as they may be generally referred) enable developers and account managers to customize the call queue experience.

The communication platform **100** of the preferred embodiment functions to provide a base service that participates in and/or facilitates communication with at least one endpoint.

The communication platform can provide a wide variety of services. The communication platform can be a network provider to a collection of endpoints. The communication platform can alternatively include a plurality of call routers such that the communication platform **100** routes calls to various endpoints and/or services according to some configured logic. Preferably, the communication platform **100** facilitates executing communication applications. Communication applications are preferably uniquely configured by developers to provide a wide variety of different solutions such as implementing call trees, conference calls, customer service call systems, automated phone message service, voicemail, programmatic messaging/notifications, or any suitable application.

The communication platform **100** is preferably multi-medium and multitenant. The multi-medium aspect of the communication platform **100** can be defined as a communication platform that facilitates communication over multiple forms of communication. The forms of communication can include PSTN, SIP, SMS, MMS, WebRTC, voice chat, video chat, screen-sharing sessions, IP based messaging, and/or other forms of communication. The communication platform **100** is preferably multitenant in the sense that multiple users, accounts, or entities share resources of the communication platform. The accounts preferably use account authentication practices when working with the application during graphical user interface configuration, API interactions, and/or during other suitable stages of account management. Accounts can further include sub-accounts or any suitable segmentation of application control and operation. The queues of an account are preferably independent from other queues and furthermore other accounts. In some variations, platform level queues may be used to manage account queues to provided resources of the communication platform. The communication platform is preferably a cloud service operating on a distributed computing system, but may alternatively be a computer cluster, an on-premise installation, or on any suitable computing infrastructure. The communication platform operates substantially similar to the communication platform described in U.S. Pat. No. 8,306,021, issued 2 Apr. 2009, which is hereby incorporated in its entirety by this reference. The communication platform may alternatively be any suitable network accessible platform.

The queue management resource no functions to control and alter the state of queue resources. The queue management resource no is preferably capable of instantiating, managing, and deleting a queue resource. When an item is instructed to be enqueued and a queue does not exist within the scope of that queue (e.g., within an account, sub-account, platform, etc.), the queue management resource preferably creates a queue resource and adds the enqueued item to the queue. Multiple queues may additionally be created within any given scope. Queues can preferably be specified by including a queue identifier in an enqueue request. An identifier for a non-existent queue preferably results in a queue with the specified queue identifier.

The items added to a queue are preferably communication sessions. The enqueued communication sessions can be added through any suitable process and do not have to originate from the same source. Some communication sessions may be added programmatically through the API, while some may be active communication sessions that transitioned to the queue to await the limited resource. An enqueued communication session can have at least two different states or modes of operation: an active state and an inactive state. The enqueued communication session may be an active communication session with a live participant on at least one endpoint of the communication. A wait-state application (e.g., queue-state application executed while waiting in the queue) will preferably execute and play audio or perform text-to-speech conversion or otherwise engage the connected endpoint as the caller waits. In another variation, the communication session is in an inactive state. An inactive communication session is preferably a proxy for a communication session. The proxy for a communication session is preferably not a live connection, but represents some entity or action that will occur upon dequeuing. The enqueued inactive communication session preferably includes a parameter specifying a communication endpoint that should be connected upon dequeuing. For example, an inactive communication session may be enqueued in response to a communication application of a received SMS message or in response to an API request. In this example, someone can get in a call waiting line and be connected when it's the entity's turn. A communication session can additionally transition between inactive and active. The queue is preferably persistent such that state or place of a communication session in the queue is maintained during changes in state of a communication session. For example, once in line, a caller can hangup (e.g., transition from active to inactive), and the queue will maintain the communication session and take appropriate action to connect the caller when the session is dequeued. A communication session proxy item is progressed through the queue, and upon dequeuing, the caller is connected.

The enqueued communication sessions can additionally be queried and/or manipulated through the API **120**. Additionally, a queued communication (active, inactive or an alternative state), may be configured with various queue-state applications. Queue-state applications are preferably executable instructions, services, playable media, and the like that are set to be initiated and performed during particular stages of the queue. A queue-state application is preferably individually executed for a single communication session. Queue-state applications can include at least the variations of an added-state application, a wait-state application, and/or an action state application. There may alternatively be similar applications that execute in the background or interject into the communication sessions based on overall state of the queue or an account. An added-state application is preferably invoked before or alternatively directly after a communication session is added to a queue. The added-state application can be used to announce to the user that they have been added to a queue. A wait-state application is preferably invoked while the communication session is waiting. The wait-state application is preferably invoked repeatedly. Alternatively, the wait-state application may be invoked when updating position or in response to any suitable event. Each time the wait-state application is invoked, updated queue statistics (e.g., estimated wait time, place in the queue, etc.) may be supplied to the application. For example, the wait-state application may include text-to-speech instructions to read to the current user their place in line and then play music for one minute before ending. When this application is looped, the caller is updated with their wait position roughly every minute. The wait-state application may additionally be invoked for non-active communication sessions. Since there is no active endpoint connected in this variation, the wait-state application functions as a background script. For example, the wait-state application may be used to send an SMS message when the user's place in the queue is updated. The action-state application (i.e., an application for the newly dequeued-state) is preferably invoked in response to the communication session being dequeued. For example, the action-state application may play a message notifying a listener that the call is being connected.

The action-state application may be configured or defined when a communication session is queued, when the queue is created, or when a dequeuing agent makes a dequeue request. A post-action application can be specified to be transferred control logic control after a dequeuer agent (e.g., a customer care representative) hangs up. Similarly, the control logic can be transferred to an original application that made the original enqueue request. For example, an initial application may include an enqueue instruction followed by instructions to receive customer feedback. A caller is first queued, then an agent helps the caller, and after the agent hangs up, the caller is transferred back to the original application so that the customer feedback instructions can execute. Any alternative types of applications may be configured or specified for the queue or for interactions with the queue.

The queue management resource no additionally includes operational configuration to manage the procession of communication sessions in a queue resource. The queue is preferably managed to enable first in first out queue behavior. The queue can alternatively enforce first in last out queue behavior. The queue could also use various prioritization policies to enable particular queued items to be move through the queue faster or slower than other items. The queue management resource preferably selects the item to dequeue, updating remaining items in the queue to reflect their new position, and add an item to the queue. The queue management resource is preferably responsive to the queue targeted API calls and particular application instructions.

The API **120** with queue targeted interfaces functions to enable programmatic interaction with queued items. An API **120** is preferably a REST API. The API preferably works according to an HTTP request and response model. HTTP requests (or any suitable request communication) to the communication platform **100** and/or the queue management resource **110** preferably observe the principles of a RESTful design. RESTful is understood in this document to describe a Representational State Transfer architecture as is known in the art. The RESTful HTTP requests are preferably stateless. The components of the communication platform and/or the interface service preferably do not need to remember or store previous communications to be aware of the state. Additionally or alternatively, the API **120** may be used or accessed through application instructions. The API **120** preferably works around a queue instance resource that allows users to query and manage the state of individual call queues. Call queues can be referenced using a queue identifier. When using a REST API, a particular queue may be referenced by a resource URI with the following pattern: "/2010-04-01/Accounts/{AccountSid}/Queues/{QueueSid}". A queue resource may include various properties such as an identifier, a friendly name (i.e., a user-provided string that identifies the queue), a current size metric, maximum size, average wait time, and/or any suitable property. A queue resource can additionally include a members sub-resource. The members sub-resource is preferably a list of communication sessions currently in the queue. A member instance is preferably the construct or proxy for an enqueued communication session. A member resource can include properties such as data enqueued, wait time, position, queue-state application configuration, and/or any suitable properties. The API **120** is preferably used to query information of the queue resources, but may additionally be used to manipulate or modify aspects of the queue. When the queue resources are used in application instructions, a call router or other suitable communication processor can add, remove, connect, and/or manage members of the queue (i.e., communication sessions of the queue). In one implementation, there is an enqueue instruction, which can be used to add a communication session as a member resource. Attributes of the enqueue instruction can include an action-state application configuration, a wait-state application configuration, other queue-state application configuration, and/or any suitable attribute. The name of the queue may additionally be specified to identify which queue is to be used. If no queue is specified, a default queue can be used. A queue-state application configuration is preferably an absolute or relative URI, but the configuration can alternatively be application logic, an application data file, or any suitable application configuration. Queue-state applications may be limited in their functionality. For example, a wait-state application may be limited to play, say, pause, hangup, redirect, leave, and gather instructions during a telephony communication. Accessing queued members preferably involves using a queue instruction. The queue instruction is preferably specified within a dial instruction. When invoking a queue instruction within a dial instruction, the queue is accessed and the first enqueued communication session is connected. If the queue is empty, one variation may include the dialing entity or agent waiting until a new communication session joins the queue. Alternatively, an error or other suitable response may be returned if the queue is empty or does not exist. An application configuration can be configured with the queue instruction to specify an application to be invoked directly preceding, during, or directly after dequeuing a communication session.

2. Method for Queuing a Communication Session

As shown in FIG. **2**, a method for queuing a communication session of a preferred embodiment includes at a queue manager receiving a request to enqueue a communication session S**110**, adding the communication session to a queue for a plurality of communication sessions S**120**, managing the procession of communication sessions in the queue S**130**, and dequeuing the communication session and bridging the communication session to a second entity according to the management of the queue S**140**. As shown in FIGS. **3**A and **3**B, the method may additionally include transferring control of the communication session to a wait-state application while the communication session is in the queue S**122** and/or transferring control of a communication session to an action-state application in response to dequeuing the communication session S**142**. The method functions to simplify management of concurrent communication sessions of an application. The queue operations are preferably atomic and transactionally safe, which can simplify the development process for developers of applications where such concurrent operations are common (e.g., telephony applications). In telephony applications, the method preferably enables application developers to simply customize the wait state experience for application-facing users. Applied to a telephony platform, the method functions to enable customized "holding" experiences for various telephony applications. The method is preferably implemented by a system substantially similar to the one described above but any suitable system may alternatively implement the method. The method is preferably used in combination with a communication platform such as one substantially similar to the telephony platform described in published U.S. Pat. No. 8,306,021, issued 2 Apr. 2009, titled "SYSTEM AND METHOD FOR PROCESSING TELEPHONY SESSIONS", but the communication platform may alternatively be any suitable communication platform. In a telephony platform or an alternative suitable platform, a multitenant and scalable resource may facilitate the operation of a plurality of queue managers. In such an embodiment, the telephony platform can preferably enable the benefits of the

method for a plurality of different applications/users on the platform. The method is preferably used with voice/telephony-based applications. Voice/telephony-based applications can include synchronous communication sessions over PSTN, SIP, or other suitable voice protocols. The method may additionally or alternatively be applied to other communication protocols such as SMS, MMS, IP messaging, video, screen sharing, and/or any suitable communication protocol. In many exemplary situations, the queue of the method is used by a telephony application when a resource (e.g., a human customer service representative) is unavailable, but the queue may be used for any suitable purpose such as for limited hardware/software resources, creating transactionally safe operations, and/or any suitable application of a queue.

Step **S110**, which recites at a queue manager receiving a request to enqueue a communication session, functions to obtain a notification from an entity to add a communication session to a queue. The request is preferably communicated from a telephony application. More preferably, the request is initiated by a telephony application instruction such as in the example request shown in FIG. **4**. The communication session is preferably an active telephony/voice call, but may alternatively be a video communication session, a screen sharing session, multi-media session, a gaming session, or any suitable communication session. The communication session preferably represents a currently active or a potentially active communication channel to an entity/user. In one example, a caller may be active on one leg of the communication session while being added to a queue. The application instructions directing logic of the communication session will preferably include an "enqueue" instruction or the equivalent instruction(s). Upon encountering the "enqueue" instruction, the communication instruction is preferably enqueued. The communication session may alternatively not include a live caller connected to the communication session and/or may be a proxy for a communication session—the communication session to enqueue may be an inactive communication session. A proxy communication session may have been programmatically queued, initiated through a asynchronous communication channel (e.g., SMS), be an active call that became inactive during the queuing process, or not have a connected live caller for any suitable reason. The request preferably specifies the account or scope of the queue. In an exemplary use case, the request specifies an account identifier and optionally an explicit queue identifier. Within a multi-tenant environment, the identifiers of the queue are used to coordinate management of multiple queues simultaneously.

Preferably, the request to enqueue a communication session specifies various queuing parameters that define the queuing behavior for that communication session. The queuing parameters may include a wait-state application, an action-state application, a queue identifier, a time-out, queue error application, and/or any suitable queuing parameter. The wait-state application, the action-state applications, and/or any suitable application parameters are preferably universal resource indicators (URIs) that reference an internet accessible resource with application instructions. The wait-state, action-state, or other applications may alternatively be any suitable reference to a queue-state application. Alternatively, queuing parameters may be specified to define queue behavior for an application, a user-account, a platform, or any suitable entity. As mentioned, the request may specify a queue identifier. The queue identifier may be for an application-based namespace or a global namespace. A default queue for a telephony application or account may be used if a queue identifier is not specified. In other words, the lack of a specified queue identifier within an account may be an implicit identifier for the default queue of the account. If a queue does not yet exist, one is preferably created. If the intended queue is unavailable due to capacity or other limitations, a new queue may be created, an error response taken, or any suitable action may be taken.

S120, adding the communication session to a queue of a plurality of communication sessions, functions to appropriately add the communication session to a queue. The queue is preferably a prioritized or ordered list of communication sessions (i.e., queue members) ordered by priority, and the recently added communication session is preferably added to the end of the communication queue. Depending on the type of queue management policy in place, the communication session may alternatively be placed in any suitable location in the queue. The entities/members in the queue are preferably persistent and may be decoupled from any associated active communication session. In other words, the communication session stored in the queue may be a proxy representative for a prior and/or subsequent communication session. In one exemplary application described in more detail below, a caller may hang-up once they are placed on hold, but their place in the queue is maintained despite the fact that the user no longer has an active communication session. In other words, an active communication session can become inactive while in the queue, and an inactive communication session can become active. An inactive communication session preferably becomes active in various ways. In one variation, the communication session becomes active by the communication platform calling out to the original endpoint of the enqueued communication session. In a second variation, the communication platform can reestablish an active communication session for an incoming communication to the original endpoint made by the endpoint associated with the enqueued communication session. For example, endpoint A calls service B. Service B queues the caller of endpoint A and plays some waiting music. The caller of endpoint A may hang up, and the place of endpoint A is managed in the queue as if the caller was still actively listening to the waiting music. Endpoint A can then call service B a second time, and the communication platform identifies that a communication session is already established between endpoint A and service B, and will reconnect the caller to the enqueued communication session. When added to the queue, the queue manager may additionally begin accounting for the communication session when measuring queue analytics. The queue analytics preferably includes statistics on overall queue properties (e.g., average wait time, number of queued communication sessions, etc.) and individual communication session queue properties (e.g., total wait time, number in line, etc.). The queue analytics is fed back to the wait state application to enable the wait state application to feed back queue information to the enqueued user.

Additionally, Step **S120** preferably includes transferring control of the communication session to a wait-state application while the communication session is in the queue as shown in FIG. **5**, which functions to manage control logic for the communication session while in the queue with a customizable application. As described above, the wait-state application is preferably specified in the queuing request. The wait-state application is preferably created or selected by the application developer, and may be designed to perform any arbitrarily complex application logic. The wait-state application may be a simple application such as a looping audio file, or may be a complex interactive application that may require user input, transfer control to other applications, or perform any arbitrarily complex logic. The wait-state application is preferably an internet accessible resource including instruc-

tions such as a telephony markup language document. The telephony communication platform implementing the method or any suitable communication platform preferably facilitates interpreting and executing the control logic of the wait-state application. Accessing the wait-state application (and similarly for other queue-state applications) preferably includes using HTTP to access the URI of the queue-state application. The queue-state applications can reside on outside network accessible resources. The returned document is preferably a document of telephony application instructions formatted into a markup language. The wait-state application is preferably implemented with a looping behavior in that the control logic may be repeated as long as the communication session is enqueued as shown in FIG. **6**. More specifically, an HTTP request to the wait-state application URI is preferably initiated each time processing of the wait-state application completes instruction execution. Since the response of the server hosting the queue-state application can dynamically change for different requests, the wait-state application can be dynamically generated to target the current context of the communication session. Current queue analytics may additionally be provided as inputs to the wait-state application (e.g., as HTTP query parameters). In one exemplary application of the queue analytics, input would be communicating a message to the user informing the user of their position in the queue. While the wait-state application can be used for controlling an active communication session, the wait-state application can additionally be used for an inactive communication session as shown in FIG. **7**. Additionally or alternatively, other applications may be transferred control of the communication session during other stages of the queue, such as when first joining the queue, when next in line to be dequeued, when a particular queue condition is met, or at any suitable time.

In a variation of a preferred embodiment, the method may include handling an error response after a failure of adding the communication session to a queue, which functions to recover from the situation where a communication session cannot be added to a queue. A communication session may not be able to be added to a queue if the queue is full, if the wait time is beyond a wait limit, or for any suitable reason or error. A failure state application is preferably transferred control of the communication session as part of the error response. A failure state application is preferably configured in a manner substantially similar to the wait-state application and action state application. The failure application is preferably transferred control of the communication session when an error response is received after attempting to add the communication session to a queue. Alternatively or additionally, the failure state application may be invoked for an error at any suitable time such as an error when dequeuing a communication session. The failure state application may provide any suitable application logic for gracefully handling an error. The type of error and any suitable error information may be passed to the failure state application. The failure state application can preferably elect to place the communication session in a second queue. A second queue may be pre-existing or generated in response to the error.

S**130**, which recites managing the procession of communication sessions in the queue, functions to update the queue based on specified or default queue heuristics. The management of the queue preferably includes adding new communication sessions to the appropriate location, updating priority/order of communication sessions, and dequeuing communication sessions. The queue is preferably configured to apply a standardized queuing heuristic. One exemplary preferred queuing policy would be a "last in, last out" policy, where queued communication sessions are dequeued in the order they are queued. However, the dequeuing and prioritization of sessions may follow any suitable heuristic. Other queuing heuristics may include "first in, last out", a priority-based policy, random selection, or any suitable heuristic to determine the servicing of the queue. The queuing heuristic may be based on properties of the queued communication sessions such as associated account IDs, phone numbers, or any suitable parameter of the communication sessions.

The queue manager may additionally enable customization of queue behavior through specifying a parameter in the queue request or through a queue API. As mentioned, the queue manager may additionally include an interface to enable dynamic queue management. The queue manager preferably has an application programming interface (API) so that an outside party may make changes to the queue and/or query status of the queue. The queue API preferably exposes various API resources, service calls, or other mechanisms that are responsive to requests to add or remove communication sessions from a queue and to requests for analytics of the call queue. The queue API may be used to retrieve queue statistics/analytics, the status of a communication session in the queue, modify the ordering of communication sessions in the queue, add or remove a communication session, alter the behavior of the queue, delete or create queues, and/or perform any suitable action. In providing a queue API interface, the method functions to enable API calls that are responsive to managing members of a queue resource and querying information of the queue. Managing members of the queue preferably includes calls that add, remove, and/or reposition communication sessions in a queue. Querying information of a queue preferably includes accessing overall queue analytics, queue member specific analytics, and/or other forms of information about the queue. The queues are preferably presented as REST API resources. For example, a queue named Foo of account Bar may be targeted in an API request through an API call to "/2010-04-01/Accounts/Bar/Queues/Foo". By issuing an HTTP GET to this queue resource, information about the queue can be retrieved. By issuing an HTTP POST to this resource, properties such as max size, queue name, and other attributes can be changed. Similarly, members of the Foo queue resource (i.e., the enqueued communication sessions) can be targeted with API calls to "/2010-04-01/Accounts/Bar/Queues/Foo/Members/". A list of members can be retrieved or specific communication sessions can be targeted by appending a communication session id or descriptor (e.g., front or back).

S**140**, which recites dequeuing the communication session and bridging the communication session to a second entity according to the management of the queue, functions to connect the communication session with another party when the communication session is selected to be dequeued. Phrased another way, block S**140** functions to establish communication of an enqueued communication session with a second entity. Bridging the communication session to a second entity can include establishing a connection between a second endpoint and the endpoint of the communication session. If the communication session is active this may simple include merging the enqueued communication session with a second communication session, where the second communication session is one established between the second entity and at least the communication platform. If the communication session is inactive, bridging may include calling out or re-establishing an active communication session with the original caller (i.e., the user associated with the enqueued communication session), and then subsequently merging the communication session with a second communication session. The

second entity is in many exemplary situations associated with a limited resource that necessitated the need of the queue. For example, the second entity may be a voice connection to a customer representative (i.e., the dequeuer) for whom an initial caller had been waiting. The second entity may alternatively be an application or device with less capacity than demand. The dequeuing of the communication session may be initiated by the second entity, but a request to dequeue a communication session may alternatively be made by any suitable party. In one embodiment, the second entity is a telephony application that dials the queue to initiate bridging with the next communication session in the queue. As shown in FIG. **8**, telephony application instructions can be used to instruct an agent or second entity to call a queue, wherein the queue is specified by a queue identifier. Alternatively, a communication session may be dequeued in response to an API call to the queue or from an intermediary application/party messaging the queue. In one variation, the API call to dequeue may be directed at the communication automatically session selected by the call queue (e.g., the communication first in line), but the API call may alternatively be directed at a specific communication session contained in the queue. The API call is preferably used in transferring the communication session to an application. The transferred application is preferably referenced by a URI in a manner similar to those described above. The application preferably includes telephony instructions that instruct a call router on controlling state and engaging a connected user. The transferred application can perform any suitable logic such as call out to an agent and bridging a communication session between the queued entity and the agent. The application could alternatively be any suitable application logic. The queue is preferably configured to be responsive to various forms of dequeueing, such as the variation of a dequeuing agent calling into the queue and bridging the two entities or the variation of transferring the communication session to an second application. The dequeued communication session is preferably one of the set of enqueued communication sessions that is next for queue servicing. Determining the next communication session is based on the procession of communication sessions in block **S130**. For example, if the queue is a first in first out queue, then the dequeued communication session is the communication queue that has been in the queue for the longest amount of time.

Additionally, step **S140** preferably includes transferring control of a communication session to an action-state application in response to dequeuing the communication session **S142** as shown in FIG. **9**, which functions to perform actions for the transition from the enqueued state to a state of being bridged with a second entity. The action-state application is preferably an application substantially similar to the wait-state application described above. The action-state application, however, is preferably selected or designed to facilitate the transition out of the queue. In one exemplary embodiment, the action-state application is designed to play the message, "Thank you for waiting. We are now connecting you" to the user connected to the communication session. In another exemplary embodiment, there may not be an active voice session for the dequeued communication session, and the action-state application may be designed to establish a voice connection with the user associated with the dequeued communication session before bridging the communication session with the second entity. The action-state application may alternatively be any suitable application. After completion of the action-state application, the selected communication session is preferably bridged with the second entity as described above and shown in FIG. **10**.

The customizable queue of the preferred embodiment may be utilized be developers to create a wide variety of queue systems. As a first example application of a preferred embodiment, the method may be employed to enable a call waiting system where callers are not required to stay on the line. A caller can preferably communicate that they would like to hold a place in line. In one variation, the user may be placed in the line due to the control logic of a telephony application with which the user has a voice connection. Once a place in the line has been established, the user may hang up while the communication session remains queued in the persistent queue. In a second variation, the user may use an alternative form of communication to initiate getting in line. For example, the user may send a text message to a specified number, that number may be directly associated with the queue or may be associated with an application that enqueues a communication session on behalf of the user. Similarly, a communication session may be automatically enqueued without direct initiation of a user. This variation functions to illustrate how an active communication session does not need to exist prior to being enqueued. As an additional feature, while the communication session is enqueued, a wait-state application may facilitate a user establishing a connection to the enqueued communication session. The user may call a number that identifies the associated queued communication session, and connects the user with the communication session in the queue as if the user never left the holding state. Similarly, the user may send a message to a phone number and in response, receive a message indicating the wait time, position in line, and/or any suitable message. When the communication session is dequeued and there is no active connection to the queued communication session, the action-state application preferably establishes a connection with the intended user. So for a user that had hung up when placed on hold, the system preferably calls the user. After the caller has an active connection then the call is preferably bridged with the intended person or device.

In a second exemplary application of a preferred embodiment, the API of a queue is preferably used to dynamically alter the waiting experience of a user. A user may have been placed in a line waiting for the next available customer representative. While waiting for the customer representative, the user may listen to a message controlled by the wait-state application. The wait-state application may include an advertisement of a way to skip ahead in the line. For example, a user may receive queue priority by listening to an advertisement, answering a survey, agreeing to be charged a fee, or through any suitable action. In response to the action of the user, the wait-state application preferably uses the queue API to alter the ordering of the communication sessions in the queue so that the user will be dequeued sooner than if they had not performed that action.

A system for an alternative embodiment preferably implements the above methods in a computer-readable medium storing computer-readable instructions. The instructions are preferably executed by computer-executable components preferably integrated with a queue manager of a telephony/communication platform. The computer-readable medium may be stored on any suitable computer readable media such as RAMs, ROMs, flash memory, EEPROMs, optical devices (CD or DVD), hard drives, floppy drives, or any suitable device. The computer-executable component is preferably a processor but the instructions may alternatively or additionally be executed by any suitable dedicated hardware device. As mentioned above, the queue manager is preferably integrated with a telephony/communication platform. More preferably the queue manager is integrated into a cloud hosted

communication platform. As part of a cloud computing infrastructure, a cluster of a queue managers preferably are preferably coordinated to manage the allocation and deallocation of queues, load balancing of communication sessions in the queues, and any suitable issues of managing a plurality of queues in a multitenant environment.

As a person skilled in the art will recognize from the previous detailed description and from the figures and claims, modifications and changes can be made to the preferred embodiments of the invention without departing from the scope of this invention defined in the following claims.

What is claimed is:

1. A method comprising: at a multitenant communication platform that includes at least one programmatic queue, each programmatic queue being configured for a platform account of a plurality of platform accounts of the communication platform, the communication platform including a programmatic queue-control interface available to each platform account associated with a programmatic queue:

establishing a communication session of a platform account in the multitenant communication platform;

through the programmatic queue-control interface, receiving an enqueue request of the platform account from a first entity, the enqueue request being a request to enqueue the communication session of the platform account in a programmatic queue of the platform account that is identified by a queue identifier specified in the enqueue request, the enqueue request specifying a wait-state application uniform resource identifier (URI) of a wait-state application for the communication session when in the queue of the platform account;

adding the communication session to the queue of the platform account, the queue being a queue of a plurality of communication sessions;

upon adding the communication session to the queue, transferring control logic to the wait-state application of the specified URI;

receiving a dequeue request initiated by a second entity; and

in response to the dequeue request, managing the procession of communication sessions in the queue; and dequeuing a communication session from the queue,

wherein a wait-state application URI is specified for each of a plurality of communication sessions enqueued in the queue of the platform account, each wait-state application URI being specified by a respective enqueue request, and wherein each wait-state application URI references executable instructions specified for the platform account,

wherein each enqueue request for the queue of the platform account is a request of the platform account, and

wherein each enqueue request received through the programmatic queue-control interface specifies at least one of:

an added-state application, the added state application including at least one of: an application that is invoked before a communication session of the enqueue request is added to the queue, and an application that is invoked directly after the communication session is added to the queue,

a wait-state application that is invoked while the communication session is waiting, and

an action-state application that is invoked in response to the communication session being dequeued.

2. The method of claim **1**, wherein dequeuing a communication session from the queue comprises selectively bridging the communication session to the second entity or transferring control logic to an application specified by the second entity.

3. The method of claim **1**, wherein the enqueue request specifies an action-state application configuration of the communication session, the method further comprising: upon dequeuing the communication session, transferring control logic to the configured action-state application.

4. The method of claim **1**, further comprising providing an application programming interface (API) to the queue as at least a subset of the programmatic queue-control interface.

5. The method of claim **4**, wherein the step of receiving an enqueue request to enqueue the communication session comprises, through the API, receiving a request to enqueue the communication session as an inactive communication session while in the queue.

6. The method of claim **4**, wherein the step of receiving a dequeue request comprises, through the API, receiving a dequeue request from an outside application.

7. The method of claim **4**, further comprising collecting analytics of the plurality of communication sessions in the queue and providing access to the analytic data through the API.

8. The method of claim **1**, wherein the received enqueue request to enqueue the communication session is initiated from instructions of an initially executed application.

9. The method of claim **8**, wherein the initial application is processing an active communication session, and the enqueued communication session is an active communication session.

10. The method of claim **1**, wherein the enqueue request further specifies an action-state application configuration as a universal resource identifier (URI) referencing executable instructions executed prior to bridging the dequeued communication session; and wherein the wait-state application URI is a reference to executable instructions executed while the communication is queued.

11. The method of claim **10**, further comprising collecting analytics of the plurality of communication sessions in the queue, and repeatedly executing the executable instructions of the wait-state application with current analytics of the queue.

12. The method of claim **1**, wherein an enqueue request without an explicit queue identifier implicitly specifies a default queue of a requesting account.

13. A method comprising: at a multitenant communication platform that includes at least one programmatic queue, each programmatic queue being configured for a platform account of a plurality of platform accounts of the communication platform, the communication platform including a programmatic queue-control interface available to each platform account associated with a programmatic queue:

establishing a communication session of a platform account in the multitenant communication platform;

through the programmatic queue-control interface, receiving an enqueue request of the platform account from a first entity, the enqueue request being a request to enqueue the communication session of the platform account in a programmatic queue of the platform account that is identified by a queue identifier specified in the enqueue request, the enqueue request specifying a wait-state application uniform resource identifier (URI) of a wait-state application for the communication session when in the queue of the platform account;

adding the communication session to the queue of the platform account, the queue being a queue of a plurality of communication sessions;

requesting wait-state application instructions from the wait-state application URI over an HTTP-based protocol, wherein analytics of the queue are included in a request to the wait-state application URI;

while in the queue, executing the wait-state application instructions;

managing the procession of communication sessions in the queue;

providing a queue application programming interface (API) that is responsive to API calls adding or removing communication sessions in a queue and API calls querying analytics of the queue, wherein the queue API is at least a subset of the programmatic queue-control interface;

receiving a dequeue request directed at the queue; and

dequeuing the communication session in response to the dequeue request,

wherein a wait-state application URI is specified for each of a plurality of communication sessions enqueued in the queue of the platform account, each wait-state application URI being specified by a respective enqueue request, and wherein each wait-state application URI references executable instructions specified for the platform account,

wherein each enqueue request for the queue of the platform account is a request of the platform account, and

wherein each enqueue request received through the programmatic queue-control interface specifies at least one of:

an added-state application, the added state application including at least one of: an application that is invoked before a communication session of the enqueue request is added to the queue, and an application that is invoked directly after the communication session is added to the queue,

a wait-state application that is invoked while the communication session is waiting, and

an action-state application that is invoked in response to the communication session being dequeued.

14. The method of claim **13**, wherein dequeueing the communication session comprises transferring control logic of the communication session to an application URI specified by a second entity.

15. The method of claim **13**, wherein the queue is a persistent queue where an enqueued communication session can be in an inactive state or an active state.

16. The method of claim **13**, further comprising measuring queue analytics; and repeating the requesting the wait-state application instructions of the wait-state application with current queue analytics.

17. The method of claim **13**, wherein the enqueue request further specifies an action-state application universal resource identifier (URI); further comprising transferring control logic of a dequeued communication session to the application of the action-state application URI prior to bridging the communication session to a second entity.

18. The method of claim **13**, wherein the enqueue request further specifies an error-state application universal resource identifier (URI); further comprising transferring control logic of a communication session to the application of the error-state application URI if an error is encountered with the queue.

19. The method of claim **13**, wherein the step of receiving an enqueue request to enqueue the communication session comprises, through the API, the enqueue request requesting to enqueue the communication session as an inactive communication session while in the queue.

20. The method of claim **13**, wherein the received request to dequeue the communication is received through a telephony instruction to call the queue.

21. The method of claim **13**, wherein the queue is persistent; and

wherein active communication sessions in the queue can become inactive while in the queue.

22. A method comprising: at a multitenant communication platform that includes at least one programmatic queue, each programmatic queue being configured for a platform account of a plurality of platform accounts of the communication platform, the communication platform including a programmatic queue-control interface available to each platform account associated with a programmatic queue:

establishing an active voice call session of a platform account in the multitenant communication platform;

through the programmatic queue-control interface, receiving an enqueue request of the platform account from a first entity, the enqueue request being a request to enqueue the voice call session of the platform account in a programmatic queue of the platform account that is identified by a queue identifier specified in the enqueue request, the enqueue request specifying a wait-state application uniform resource identifier (URI) of a wait-state application for the active voice call session when in the queue of the platform account, the enqueue request being initiated by an application instruction of the voice call session;

adding the voice call session to the queue of the platform account, the queue being a queue of a plurality of voice call sessions;

collecting queue analytics;

upon adding the voice call session to the queue, transferring control logic of the voice call session to the wait-state application as specified by the wait-state application URI, wherein control logic of the wait-state application is looped with current queue analytic input while waiting in the queue;

receiving an instruction to call the queue and dequeue a voice call session of the queue, the instruction specifying an action-state application universal resource identifier (URI) of an action-state application;

in response to the instruction to call the queue, managing the procession of voice call sessions in the queue and dequeuing a voice call session from the queue;

upon dequeuing the selected voice call session, transferring control logic to the action-state application specified by the action-state application URI; and

bridging a voice call of the selected voice call session to a second entity,

wherein a wait-state application URI is specified for each of a plurality of voice call sessions enqueued in the queue of the platform account, each wait-state application URI being specified by a respective enqueue request, and wherein each wait-state application URI references executable instructions specified for the platform account,

wherein each enqueue request for the queue of the platform account is a request of the platform account, and

wherein each enqueue request received through the programmatic queue-control interface specifies at least one of:

an added-state application, the added state application including at least one of: an application that is invoked before a voice call session of the enqueue request is

added to the queue, and an application that is invoked directly after the voice call session is added to the queue,

a wait-state application that is invoked while the voice call session is waiting, and

an action-state application that is invoked in response to the voice call session being dequeued.

23. The method of claim **1**, wherein transferring control logic to the wait-state application comprises transmitting a first HTTP request with queue analytics to the wait-state application URI, receiving instructions as part of a response to the first HTTP request, and processing the instructions in controlling logic of the voice call session.

24. A method comprising: at a multitenant communication platform that includes at least one programmatic queue, each programmatic queue being configured for a platform account of a plurality of platform accounts of the communication platform, the communication platform including a programmatic queue-control interface available to each platform account associated with a programmatic queue:

establishing a communication session of a platform account in the multitenant communication platform;

through the programmatic queue-control interface, receiving an enqueue request of the platform account from a first entity, the enqueue request being a request to enqueue the communication session of the platform account in a programmatic queue of the platform account that is identified by a queue identifier specified in the enqueue request, the enqueue request specifying wait-state application configuration for the communication session when in the queue of the platform account;

adding the communication session to the queue of the platform account, the queue being a queue of a plurality of communication sessions;

upon adding the communication session to the queue, the communication platform transferring control logic according to the specified wait-state application configuration;

receiving a dequeue request initiated by a second entity; and

in response to the dequeue request, managing the procession of communication sessions in the queue; and dequeuing a communication session from the queue,

wherein wait-state application configuraiton is specified for each of a plurality of communication sessions enqueued in the queue of the platform account, each wait-state application configuraiton being specified by a respective enqueue request, and wherein each wait-state application configuraiton references executable instructions specified for the platform account,

wherein each enqueue request for the queue of the platform account is a request of the platform account,

wherein each enqueue request of a platform account is provided by a system of the platform account external to the communication platform, and

wherein each enqueue request received through the programmatic queue-control interface specifies at least one of:

an added-state application, the added state application including at least one of: an application that is invoked before a communication session of the enqueue request is added to the queue, and an application that is invoked directly after the communication session is added to the queue,

a wait-state application that is invoked while the communication session is waiting, and

an action-state application that is invoked in response to the communication session being dequeued.

25. The method of claim **24**, wherein each wait-state application configuration is specified by a corresponding external system.

26. The method of claim **24**, wherein different wait-state application configuration is specified for each of a plurality of communication sessions enqueued in the queue identified by the specified queue identifier.

27. The method of claim **24**, wherein different wait-state application configuration is specified for each of a plurality of communication sessions enqueued in at least one queue of the multitenant communication platform.

28. The method of claim **24**, wherein the enqueue request specifies the account of the communication platform.

29. The method of claim **24**, wherein at least one external system includes an application server external to the communication platform, the application server being an application server of an application of the associated platform account of the communication platform.

30. The method of claim **24**, wherein the communication platform transferring control logic according to the specified wait-state application configuration comprises: the communication platform requesting the wait-state application executable instructions from a wait-state application URI of the wait-state application configuration; and the communication platform executing the wait-state application executable instructions.

31. The method of claim **13**, wherein analytics include at least one of statistics on overall queue properties, individual communication session queue properties, wherein statistics on overall queue properties includes at least one of average wait time and number of queued communication sessions, and wherein individual communication session queue properties include at least one of total wait time and number in line.

* * * * *